ZF Linux Devices * 1052 Elwell Court, Palo Alto, CA 94303 * Tel: 650-965-3800 * Fax 650-965-4050

ZF Linux Devices * 1052 Elwell Court, Palo Alto, CA 94303 * Tel: 650-965-3800 * Fax 650-965-4050

1. Introduction to the MachZ BIOS

The MachZ BIOS is the Phoenix 4.0 Revision 6 BIOS customized for the MachZ™, and is licensed for use with the ZF Linux Devices, Inc. MachZ System-On-a-Chip.

This manual is a supplement to the, “PhoenixBIOS™ 4.0 Revision 6 User's Manual” dated June 22, 2000 and included with the MachZ BIOS Release Set 1.00. It covers MachZ specific configuration settings and utilities used to manage the MachZ BIOS.

Certain Hypertext Links in this file will take you either to the web, or to other ZF Linux Devices documents. For the document links to work, the PDF version of this file should be in the same directory as the other files. On the MachZ Integrated Development System CD, all the PDF files are in subdirectory \documents.

See the PhoenixBIOS™ 4.0 Rev6 User Manual.PDF.

2. Features

In addition to the standard features documented in the PhoenixBIOS™ User's Manual, the MachZ BIOS includes these extended features important for embedded applications.

- ZFlash OS Loader - Built in hook that enables Operating Systems such as Linux and VxWorks to boot from the same flash chip which contains the BIOS.
- ZFlash legacy ISA extension processor - allows user extension ROMs to be placed in the same flash device as BIOS.
- Configuration settings that manage MachZ ZF Logic Memory and I/O Chip Selects for Disk On Chip, flash based extensions and custom I/O hardware.
- Advanced Power Management 1.2 Functions.
- Universal Serial BUS Host Controller and Legacy Configuration Settings.
- Infrared support.
- Watchdog Timer Function
- Remote Management from PC Host
- Resident Flash Disk Function

2.1. MACHZ BIOS SET

The MachZ BIOS Set is composed of the items in the following list. Release 1.00 of the set can be obtained from us using ZF part number: 9270-0012-010000

Zip File, MZPB1_00.ZIP contains the components of release 1.00 of the MachZ BIOS Set. The Zip File contents are:

- PhoenixBIOS™ 4.0 revision 6 User's manual
- Machz BIOS User's Manual Supplement
- README.TXT - release notes
- README.PDF - release notes in PDF format.
- MZPB1_00.ROM binary image file with Linux Devices, Inc. Splash Screen
- MZPB1_00.RON binary image file - no splash screen.
- MZPB1_00.RED binary image file - Universal Console Redirect.
- MZPB1_00ROM.BIN Z-tag Manager binary image file with Linux Devices, Inc. Splash Screen (AMD29F0XX 8-BIT FLASH)
- MZPB1_00RON.BIN Z-tag Manager binary image file - no splash screen.(AMD29F0XX 8-BIT FLASH)
- MZPB1_00RED.BIN Z-tag Manager binary image file - Universal Console Redirect (AMD29F0XX 8-BIT FLASH)
- AMDFLASH.EXE utility
- MZPB1_00ROM_IA.BIN Z-tag Manager binary image file with Linux Devices, Inc. Splash Screen (StrataFlash)
- MZPB1_00RON_IA.BIN Z-tag Manager binary image file - no splash screen. (StrataFlash)
- MZPB1_00RED_IA.BIN Z-tag Manager binary image file - Universal Console Redirect (StrataFlash)

3. Installation

The MachZ BIOS binary image files are 256 Kbytes in length. Any of these files may be placed in an ROM, EPROM or Flash Device that is chip selectable by the processor's reset vector, (0FFFFFFF0h) and chip select 0. Although the image is 256 Kbytes in this space, during initialization, certain blocks are decompressed and/or discarded. The result is that a 128 Kbyte image is loaded into the shadow area of system memory at 0E0000h to 0FFFFFh. The BIOS image may be loaded into the Integrated Development System AMD Flash using either the Z-tag Dongle or from a DOS prompt using the AMD-FLASH.EXE Utility.

Optionally, external Programmers may be used to transfer the BIOS image into EPROM or Flash devices.

3.1. Installation Using AMDFLASH.EXE

AMDFLASH is a convenient way to update the BIOS in the Integrated Development System or other Target Board which has a 2 MB AMDFLASH chip installed. AMDFLASH supports only AMD flash chips, and requires that you have a working BIOS already installed (and thus can boot DOS).

You can use AMDFLASH from floppy or hard disk. AMDFLASH.EXE always loads the BIOS image from a file called, BIOS.ROM, residing on a floppy or hard disk

1. Use COPY /B to transfer the desired BIOS image to the BIOS.ROM file. This file must be in the same directory as AMDFLASH.EXE. Example:

COPY /B MZPB1_00.ROM BIOS.ROM

2. Set the Jumpers so that it boots from the AMDFLASH (they may be set that way already - the newer systems are set this way). Jumper pins 4 and 6 of JP7 and switch S3 #12 is OFF.
3. Since the AMDFLASH utility runs from a DOS prompt, make sure you are able to boot with any working BIOS, the GS from the ATMEL CHIP or a previous version of Phoenix from the AMD Flash. For example, if you boot DOS from the GS/ATMEL then change the CS0 jumper from pins 3-5 to pins 4-6 (same position in the opposite column) and place switch #12 of S3 to the right before executing the AMDFLASH utility.
4. Boot to DOS. You can run DOS from the floppy disk or the hard disk.
5. Copy program AMDFLASH.EXE (attached in the ZIP file) to the DOS disk. Copy BIOS.ROM (the BIOS image) to the same directory. For example: COPY /B mzpb_b01.red BIOS.ROM (always use the /B binary argument if you COPY from a DOS prompt.)
6. Run, “AMDFLASH 0". It will post status messages on the display.[1]

Do not change any jumpers unless you booted from the ATMEL Flash (see step 2 above). This version of the program will put 1 copy of the BIOS.ROM in the IDS AMD flash.

1. If you have trouble, running AMDFLASH 0. We recommend removing any ISA slot boards <except video>. Currently AMDFLASH is “hardwired” for a 2 MB Flash Chip.

3.2. BIOS Install using a Dongle

The Z-tag Manager software application may be used to load the BIOS image into the Z-tag Dongle. The Dongle can then be used load the BIOS on systems that support the Z-tag interface such as the IDS. See the examples in Chapter 4 of the MachZ Integrated Development System Quick Start Guide.[2]

The Dongle must be loaded with a second SEEPROM. To obtain this part contact ZF support and ask for Part Number 3100-0165-00. Dongles manufactured after January, 2001 already have this update. The Z-tag Manager will provide an error message if the Dongle does not have enough SEEPROM.

4. BIOS Setup

4.1. Main

For the screens below, see the PhoenixBIOS™ 4.0 Rev6 User Manual.PDF.

- Primary Master
- Primary Slave
- Secondary Master
- Secondary Slave
- Memory Cache
- Boot Options
- Keyboard Features

2.The flash programmer used by the Dongle will let you specify where in the flash you wish the BIOS placed. It should be in the high addresses of the flash. Since the MachZ Phoenix BIOS is a 256K image, use starting address 1C0000 if you have a 2MB Flash, and C0000 if you have a 1 MB flash (etc.).

4.2. Advanced

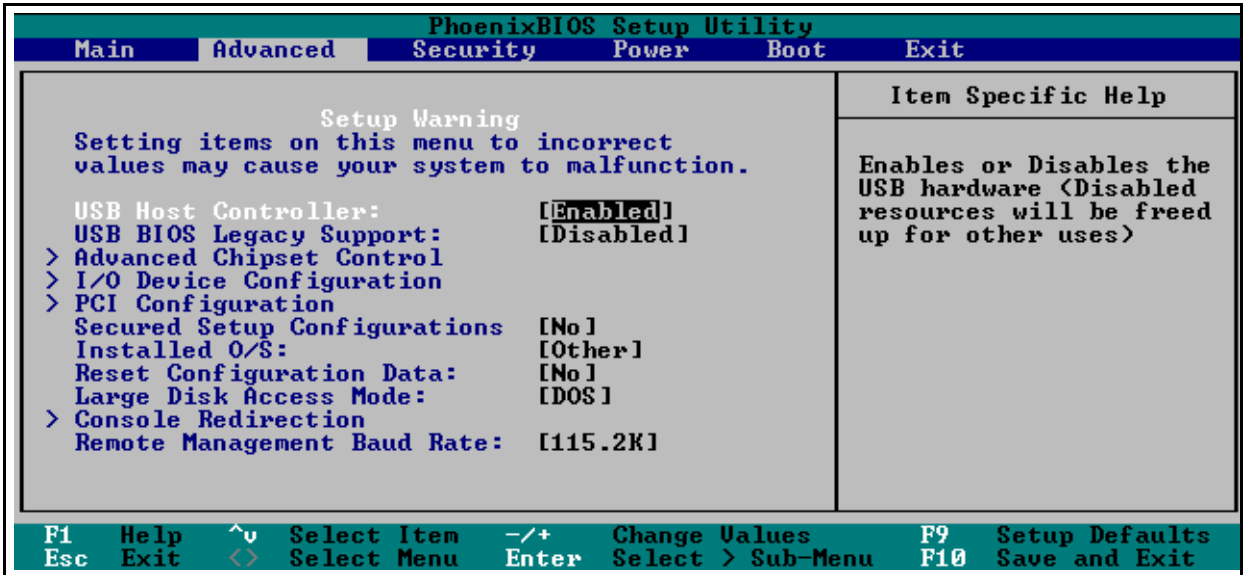


Figure 1. Advanced Setup Screen

Table 1: Advanced Setup Screen

Feature	Options	Description
USB Host controller	Disabled **Enabled**	Enables or Disables the USB hardware (Disabled resources will be freed up for other uses)
USB BIOS Legacy Support	**Disabled** Enabled	Enables or Disables support for USB Keyboards and Mice. (Enable for use with a non-USB aware Operating System such as DOS or UNIX)
Advanced Chipset Control		See 'Advanced Chipset Control' on page 7
I/O Device Configuration		See 'I/O Device Configuration' on page 11
PCI Configuration		See 'PCI Configuration' on page 14
Secured Setup Configurations	**No** Yes	'Yes' prevents a Plug and Play Operating System from changing system settings.

Table 1: Advanced Setup Screen

Installed O/S	**Other** Win95	Select the operating system installed on your system which you will use most commonly. Note: An incorrect setting can cause some operating systems to display unexpected behavior.
Reset Configuration Data	**No** Yes	Select 'Yes' if you want to clear the Extended System Configuration Data (ECSD) area.
Large Disk Access Mode	Other **DOS**	UNIX, Novel NetWare, or other operating systems, select 'Other'. If you are installing new software and the drive fails, change this selection and try again. Different operating systems require different representations of drive geometries.
Console Redirection		See 'Console Redirection' on page 19
Remote Management Baud Rate	**115**.2K 57.6K 38.4K 28.8K 19.2K 14.4K 9600 2400	Selects the baud rate used for serial remote configuration mode.

4.3. Advanced Chipset Control

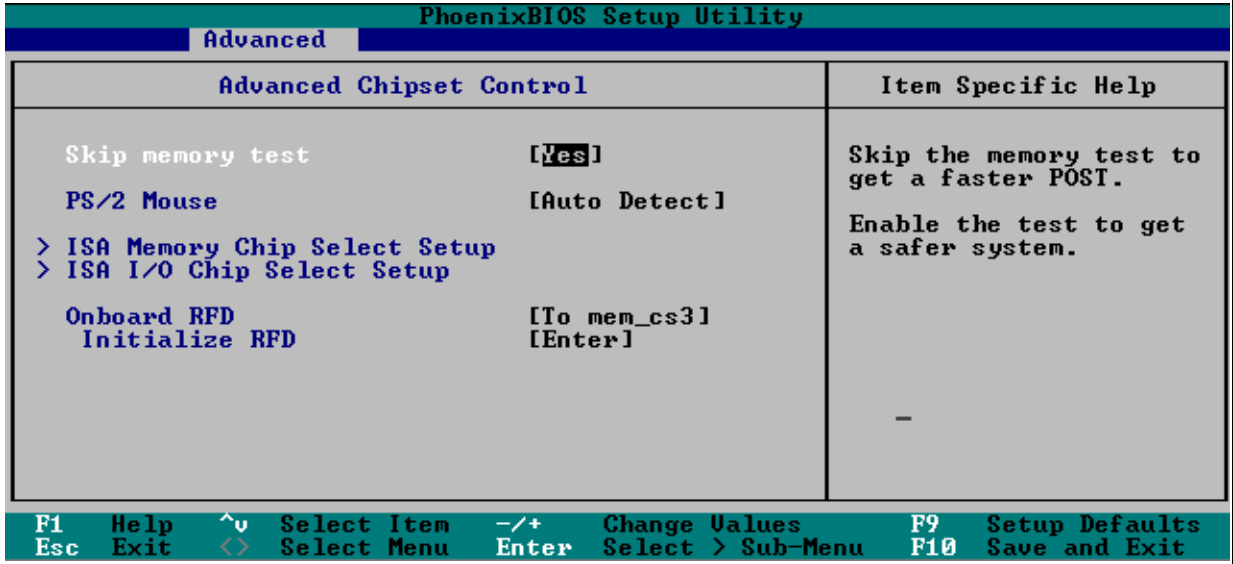


Figure 2. Advanced Chipset Control

Table 2: Advanced Chipset Control

Feature	Options	Description
Skip memory test	No **Yes**	Skip the memory test to get a faster POST. Enable the test to get a safer system.
PS/2 Mouse	Disabled Enabled **Auto Detect**	Disabled' prevents any installed PS/2 mouse from functioning, but frees up IRQ 12. 'Enabled' forces the PS/2 mouse port to be enabled regardless if a mouse is present. 'Auto Detect' will enable the PS/2 mouse only if present. 'OS Controlled' only displayed if the OS controls the mouse.
ISA Memory Chip Select Setup		See 'ISA Memory Chip Select' on page 8
ISA I/O Chip Select Setup		See 'ISA I/O Chip Select Setup' on page 10

Table 2: Advanced Chipset Control

Onboard RFD	Disabled To mem_cs1 To mem_cs2 **To mem_cs3**	Selects whether the onboard flash disk is enabled.
Initialize RFD	Enter	Initializes the RFD to be used as a disk emulator. WARNING! WARNING! Initialization erases all information from the RFD!

4.3.1. ISA Memory Chip Select

The Memory Chip Selects provide initial values for the four memory windows which may be created using the ZF-logic built in the MachZ chip. For more information, see 'Understanding Memory Windows' on page 21.

note: Version 1.00 requires all hex values to be converted to their decimal equivalent. The three digit hex fields must be input in decimal -- so FFF would be 4095. See Table 13 on page 28 for an example.

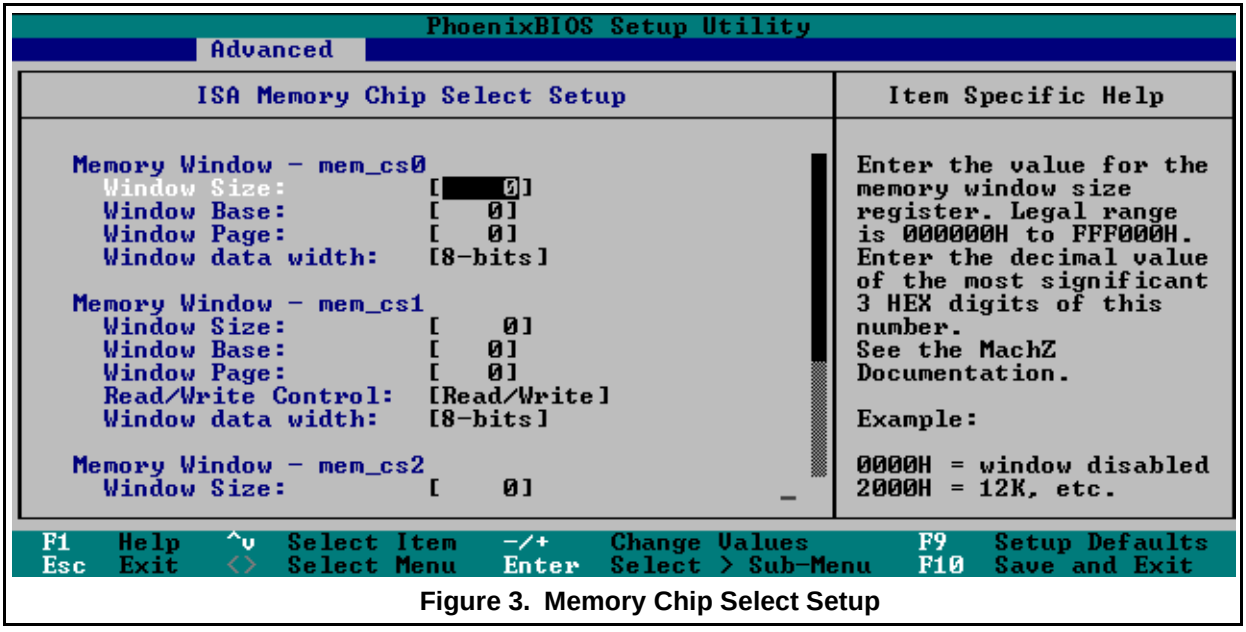


Figure 3. Memory Chip Select Setup

Table 3: ISA Memory Chip select setup

Feature	Options	Description
Window Size		Enter the value for the memory window size register. Legal range is 000000H to FFF000H. Enter the decimal value of the most significant 3 HEX digits of this number See the MachZ Documentation. Example: 0000H = window = disabled, 1000H = 8K, 2000 = 12K, etc.
Window Base		Enter the value for the memory window base register. Legal range is 000000H to FFF000H. Enter the decimal value of the most significant 3 HEX digits of this number See the MachZ Documentation. Example: 0C0000 = base address is 000C0000 (or C000:0)
Window Page		Page = 1000000 - BASE + FLASHA If Base = 0D0000 set PAGE to F30000 so that D000:0 goes to address 0 in the flash. That is 1000000 - D0000 + 0. For D0000 to go to D0000 in the flash, set PAGE to 0. That is 1000000 - D0000 + D0000 (You only specify 4 digits).
Read/Write Control	**Read/Write** Read-only	Select the behavior of the memory range between read/write or read-only access type.
Window data width	16-bits **8-bits**	Select the window datapath width. The data width may be 8-bits or 16-bits.

4.3.2. ISA I/O Chip Select Setup

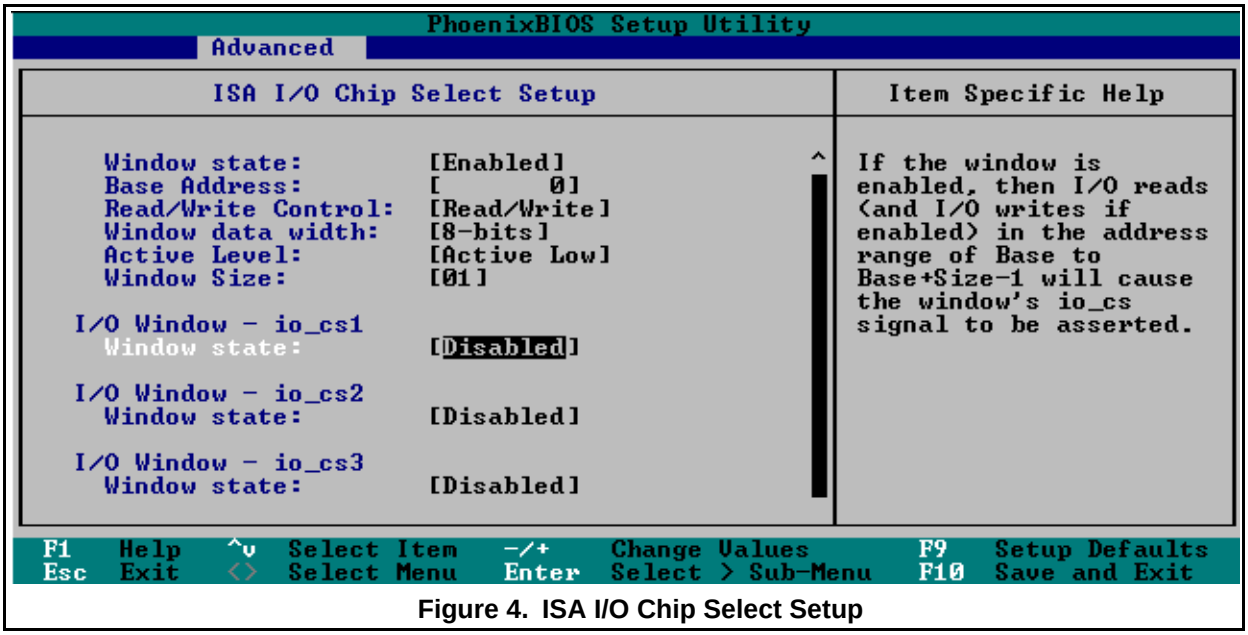


Figure 4. ISA I/O Chip Select Setup

Table 4: ISA I/O Chip Select Setup

Feature	Options	Description
Window state:	Enabled **Disabled**	If the window is enabled, then I/O reads (and I/O writes if enabled) in the address range of Base to Base+Size -1 will cause the window's io_cs signal to be asserted.
Base address:	**0** - 65535	Enter the base address for the I/O window. This address is in the range of 0 - 65535.
Read/Write Control:	**Read/Write** Read-only	Select the behavior of the I/O range between read/write or read only access type ports. Setting window to read-only mode disables the IOW_N signal on ISA bus for I/O window address range.
Window data width:	**8-bits** 16-bits	Select the window datapath width. The data width may be 8-bits or 16-bits.
Active level:	**Active Low** Active High	Select the level to assert on the io_cs pin. The MachZ asserts the selected level on the window's io_cs pin when the program accesses I/O in the window range of Base to Base+Size-1.

Table 4: ISA I/O Chip Select Setup

Window Size:	**01** - 16	Select the number of consecutive I/O address to decode starting from the I/O window base. For example, a value of 4 will decode 4 consecutive 8-bit I/O addresses, or 2 consecutive 16-bit I/O addresses. Decode will occur on I/O read, and may occur on I/O write

4.3.3. I/O Device Configuration

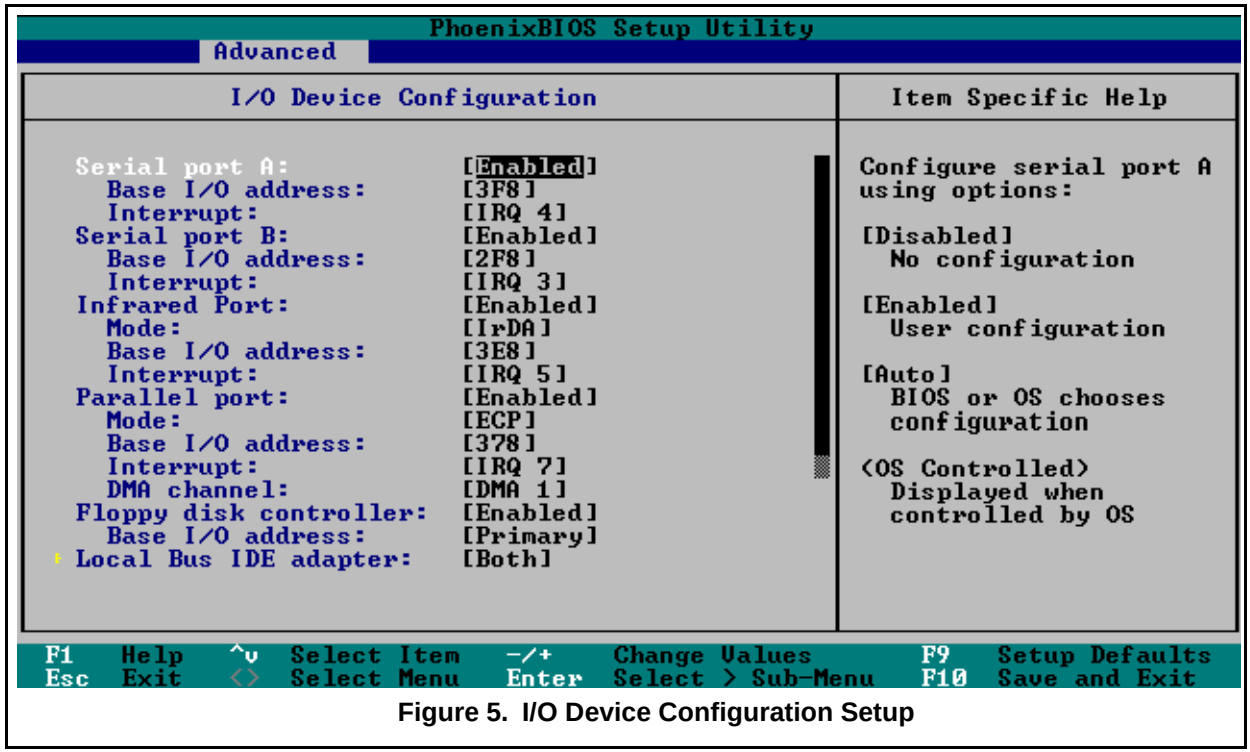


Figure 5. I/O Device Configuration Setup

Table 5: I/O Device Configuration Setup

Feature	Options	Description

Table 5: I/O Device Configuration Setup

Serial Port A:	Disabled Enabled Auto **OS Controlled**	Configure serial port A using options: [Disabled] No configuration [Enabled] User Configuration [Auto] BIOS or OS chooses configuration (OS Controlled) Displayed when controlled by OS
Base I/O address:	**3F8** 2F8 3E8 2E8	Set the base I/O address for serial port A.
Interrupt	IRQ 3 **IRQ 4**	Set the interrupt for serial port A.
Serial Port B:	Disabled Enabled Auto **OS Controlled**	Configure serial port B using options: [Disabled] No configuration [Enabled] User Configuration [Auto] BIOS or OS chooses configuration (OS Controlled) Displayed when controlled by OS
Base I/O address:	3F8 **2F8** 3E8 2E8	Set the base I/O address for serial port B.
Interrupt	**IRQ 3** IRQ 4	Set the interrupt for serial port B.
Infrared Port:	Disabled Enabled Auto **OS Controlled**	Configure Infrared port using options: [Disabled] No configuration [Enabled] User Configuration [Auto] BIOS or OS chooses configuration (OS Controlled) Displayed when controlled by OS
Mode	**IrDA** FIR	Set mode for Infrared port.
Base I/O address:	3F8 **3E8**	Select the base I/O address for Infrared port.
Interrupt	IRQ 3 **IRQ 5**	Select the interrupt for the Infrared port.
Parallel port:	Disabled Enabled Auto **OS Controlled**	Configure parallel port using options: [Disabled] No configuration [Enabled] User Configuration [Auto] BIOS or OS chooses configuration (OS Controlled) Displayed when controlled by OS

Table 5: I/O Device Configuration Setup

Mode	Output only Bi-directional EPP **ECP**	Set the mode for the parallel port using options: Output only, Bi-directional, EPP, ECP
Base I/O address:	**378** 278 3BC	Set the base I/O address for the parallel port.
Interrupt	IRQ 5 **IRQ 7**	Set the interrupt for the parallel port.
DMA channel	**DMA 1** DMA 3	Set the DMA channel for the parallel port.
Floppy disk controller	Disabled **Enabled** Auto	Configure using options: [Disabled] No configuration [Enabled] User Configuration [Auto] BIOS or OS chooses configuration (OS Controlled) Displayed when controlled by OS
Base I/O address	**Primary** Secondary	Set the base I/O address for the floppy disk controller using options: Primary, Secondary
Local Bus IDE adapter:	Disabled Primary Secondary **Both**	Enable the integrated local bus IDE adapter

4.3.4. PCI Configuration

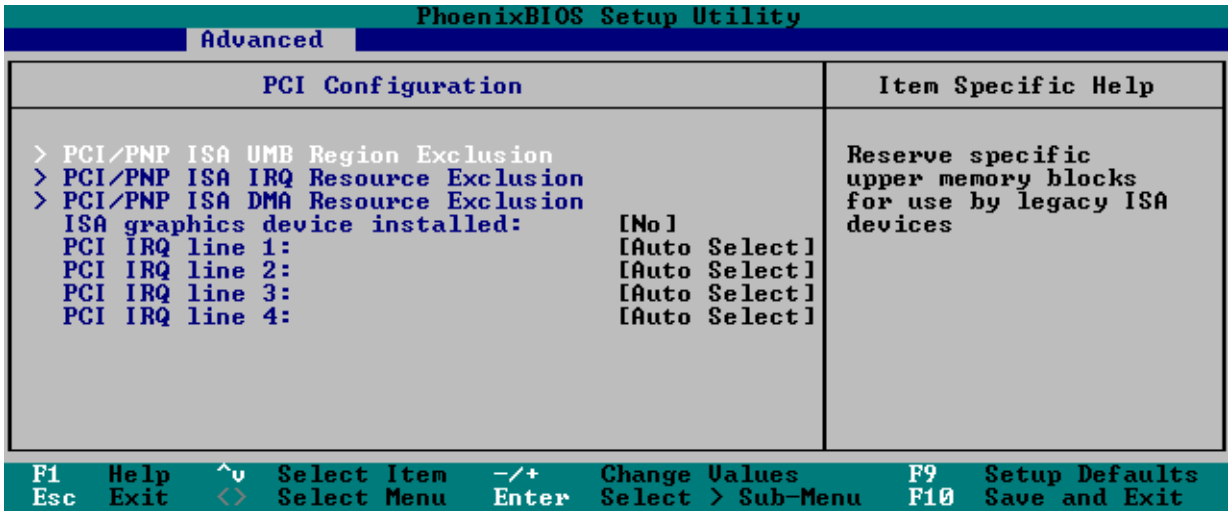


Figure 6. PCI Configuration

Table 6: PCI Configuration

Feature	Options	Description
PCI/PNP ISA UMB Region Exclusion		Reserve specific upper memory blocks for use by legacy ISA devices. See menu XXX.X
PCI/PNP ISA IRQ Resource Exclusion		Reserve specific IRQ's for use by legacy ISA devices See menu XXX.X
PCI/PNP ISA DMA Resource Exclusion		Reserve specific DMA channels for use by legacy ISA devices. See menu XXX.X
ISA graphics device installed:	**No** Yes	Enable ISA (non-VGA) graphics device to access palette data in PCI VGA device.

Table 6: PCI Configuration

PCI IRQ line 1: PCI IRQ line 2: PCI IRQ line 3: PCI IRQ line 4:	Disabled **Auto Select** 3 4 5 7 9 10 11 12 14 15	PCI devices can use hardware interrupts called IRQ's. A PCI device cannot use IRQ's already in use by ISA Enable ISA (non-VGA) graphics device to access

4.3.5. PCI/PNP ISA UMB Region Exclusion

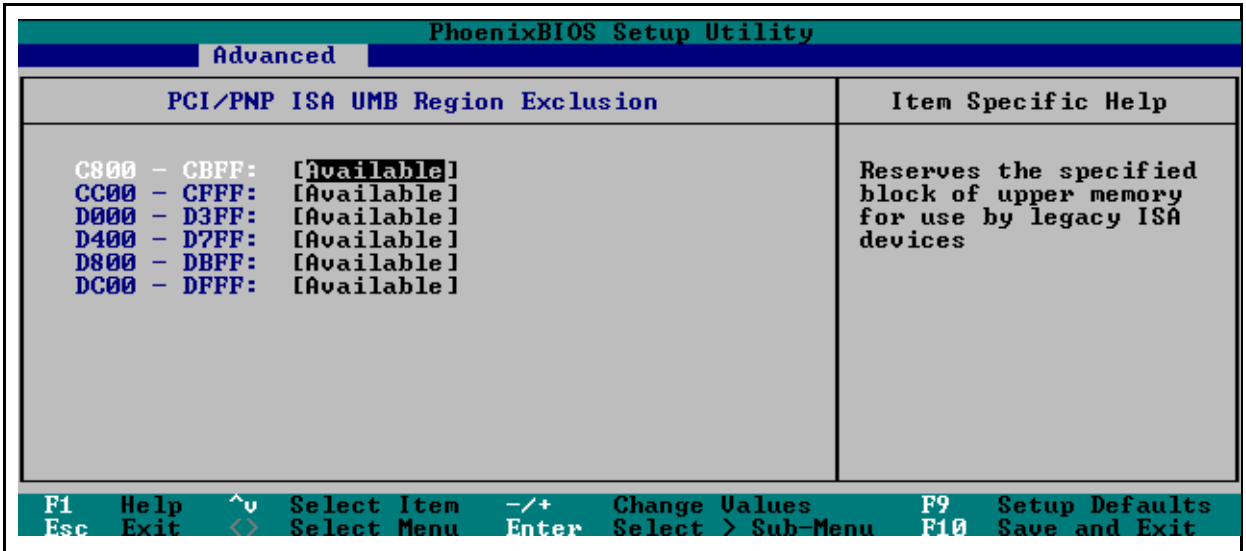


Figure 7. PCI/PNP ISA UMB Region

Table 7: PCI/PNP ISA UMB Region Exclusion

Feature	Options	Description
C800 - CBFF: CC00 - CFFF: D000 - D3FF: D400 - D7FF: D800 - DBFF: DC00 - DFFF:	**Available** Reserved	Reserves the specified block of upper memory for use by legacy ISA devices

4.3.6. PCI/PNP ISA IRQ Resource Exclusion

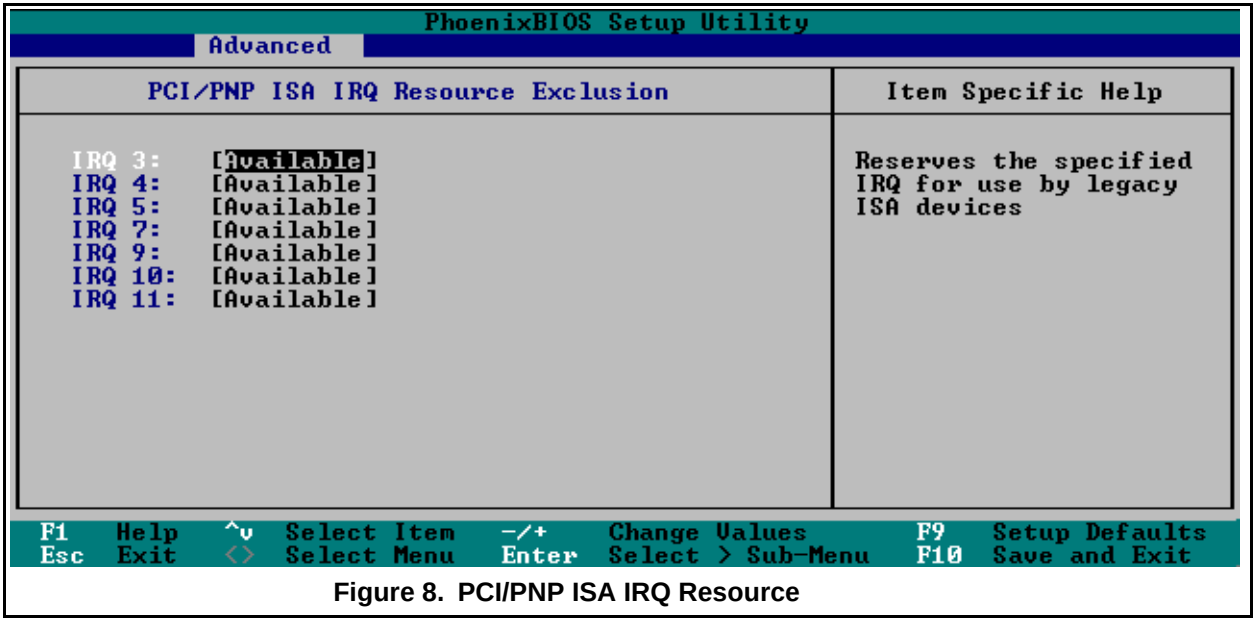


Figure 8. PCI/PNP ISA IRQ Resource

Table 8: PCI/PNP ISA IRQ Resource Exclusion

Feature	Options	Description
IRQ 3: IRQ 4: IRQ 5: IRQ 7: IRQ 9: IRQ 10: IRQ 11:	**Available** Reserved	Reserves the specified IRQ for use by legacy ISA devices

4.3.7. PCI/PNP ISA DMA Resource Exclusion

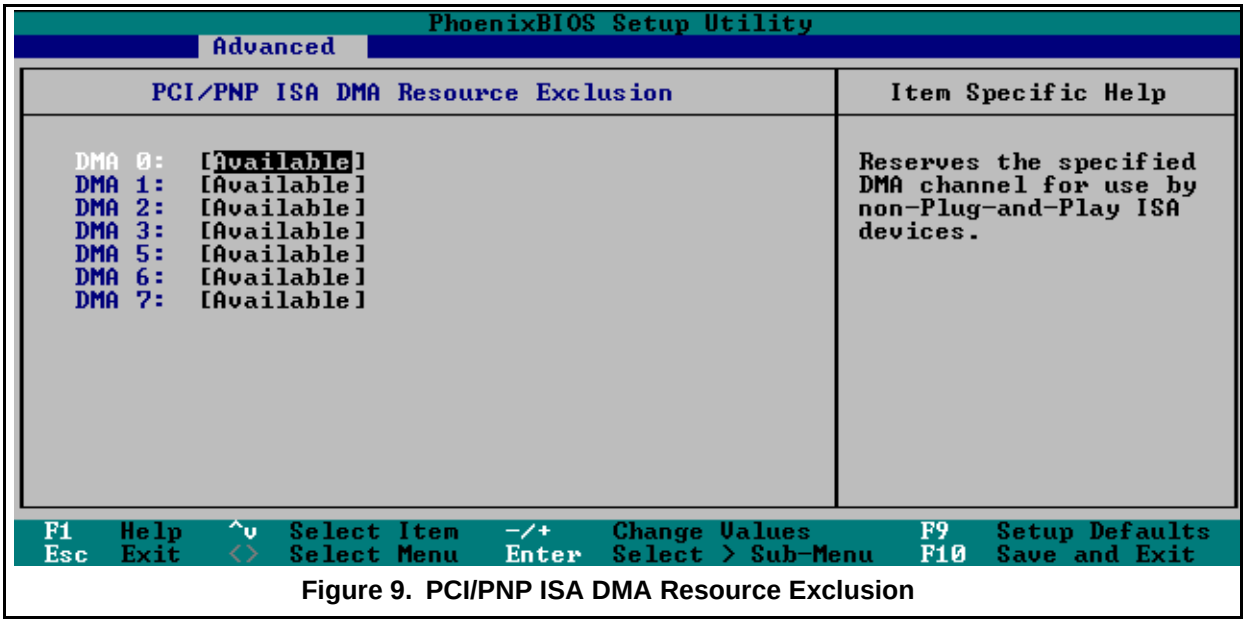


Figure 9. PCI/PNP ISA DMA Resource Exclusion

Table 9: PCI/PNP ISA DMA Resource Exclusion

Feature	Options	Description
DMA 0: DMA 1: DMA 2: DMA 3: DMA 4: DMA 5: DMA 6: DMA 7:	**Available** Reserved	Reserves the specified DMA channel for use by non-Plug-and-Play ISA devices.

4.3.8. Console Redirection

UCR (Universal Console Redirect) feature supports those embedded systems which do not (or will not) have a console (keyboard and monitor). The BIOS Binary Image file, MZPB1_00.RED, is part of the MachZ BIOS Set and is equivalent to the BIOS binary image file in MZPB1_00.ROM except the redirect setting is on by default.

A null modem cable connection is required between the MachZ COM A (COM1) and a PC/ANSI terminal or terminal emulator such as Procomm™ or Hyperterminal™. Only text mode video operations are supported during console redirect.

The default settings for the .RED (redirect) file are shown in Table 11.

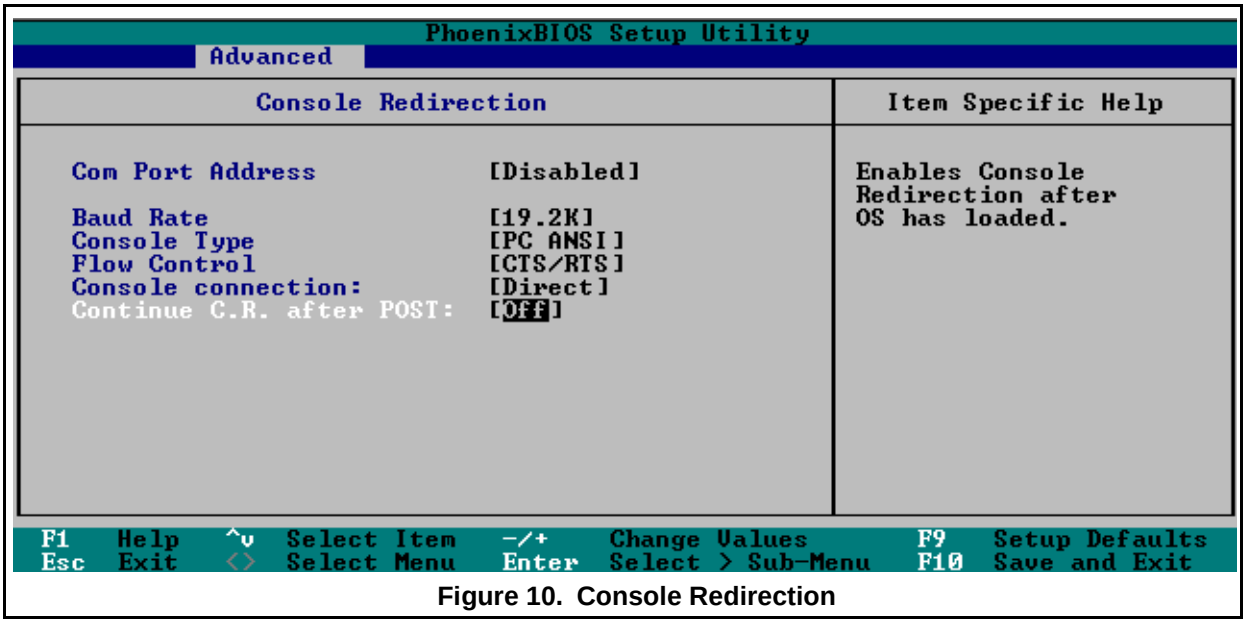


Figure 10. Console Redirection

Table 10: Console Redirection

Feature	Options	Description
Com Port Address	**Disabled** On-board COM A On-board COM B	If enabled, it will use a port on the motherboard.

Table 10: Console Redirection

Baud Rate	300 1200 2400 9600 **19.2 K** 38.4 K 57.6 K 115.2 K	Enables the specified baud rate.
Console Type	**PC ANSI** VT100	Enables the specified console type.
Flow Control	None XON/XOFF **CTS/RTS**	Enables Flow Control
Console connection:	**Direct** Via modem	Indicate whether the console is connected directly to the system or a modem is used to connect.
Continue C.R. after POST:	**Off** On	Enables Console Redirection after OS has loaded.

Table 11: Console Redirection Defaults in .RED file

Feature	Options	Description
Com Port Address	**COM A**	If enabled, it will use a port on the motherboard.
Baud Rate	**19.2 K**	Enables the specified baud rate.
Console Type	**PC ANSI**	Enables the specified console type.
Flow Control	**CTS/RTS**	Enables Flow Control
Console connection:	**Direct**	Indicate whether the console is connected directly to the system or a modem is used to connect.
Continue C.R. after POST:	**On**	Enables Console Redirection after OS has loaded.

4.4. Other Setup Screens

For the screens below, refer to PhoenixBIOS™ 4.0 Rev6 User Manual.PDF which is supplied on the ZF CD.

- Security
- Power
- Boot
- Exit
- Help

5. ZFlash OS LOADER

This feature allows Operating Systems, such as Linux or VxWorks to load and boot from the same flash device that holds the BIOS.

Just before a boot attempt of standard media devices, the MachZ BIOS scans external Flash devices address blocks. The blocks are defined by the user configurable Memory Window Chip Selects and contain a standard legacy ISA extension ROM header. A checksum (modulo 100h) is performed and if that is successful, BIOS will transfer the boot attempt to code beginning at byte 3 of the special header. When the transfer is executed, a signature parameter value of 'MORX' is passed in register EDX, allowing the user to 'authenticate' the call.

Thus, user supplied external code may be installed in flash and can continue the boot function using it's own algorithms. Details are available from ZF Linux Devices. Ask support for the, “ZFlash OS Loader Application Note Part Number 9150-0012-000”.

6. Technical Tips

In this section we will look at items and techniques particular to your effective use of the MachZ BIOS.

6.1. Understanding Memory Windows

Memory windows are a feature of the ZF Logic. . Memory windows provide chip select and addressing for SRAM and Flash Chips external to the MachZ chip.

There are two benefits of memory windows: (1) they allow interconnect to Flash and SRAM chips without extra glue logic; and (2) they allow addressing of up to 64 MB of external Flash/SRAM (4 chip selects times 16 MB per chip select) through an aperture in

the ISA address space called a memory window.

6.1.1. Memory Window Basics

Back around the time of the XT computer, a paged memory scheme was developed to allow the XT to access memory above 1 MB. The method used is shown in Figure 11 on page 23. A small aperture below 1 MB was used to address perhaps 1 to 8 MB of additional RAM through page switching.

The memory windows on the MachZ work in a similar manner. If you run program AMDFLASH on the MachZ, and specify option P (start AMDFLASH with no parameters, and then specify option P), you will see a printout similar to:

```
Window OK at C8000-CBFFFH
Window OK at CC000-CFFFFH
Window OK at D0000-
```

This specifies where you can place windows (apertures) in the space below 1 MB. Actually, it specifies memory in the range of C0000 to FFFFF which is not shadowed. The MachZ BIOS lives in the upper 128K of the first megabyte -- addresses E0000 through F0000. To make the BIOS run faster, rather than read it from 8-bit wide flash, the BIOS is copied into (typically) 32-bit wide SDRAM and then the SHADRC and SHADWC registers in the MachZ North Bridge hardware are set to enable read from the SDRAM and not to enable Write to the SDRAM. The result is that memory reads directed to this address space are directed to the north bridge SDRAM controller, which in turn is generally connected to high performance 32-bit wide SDRAM.

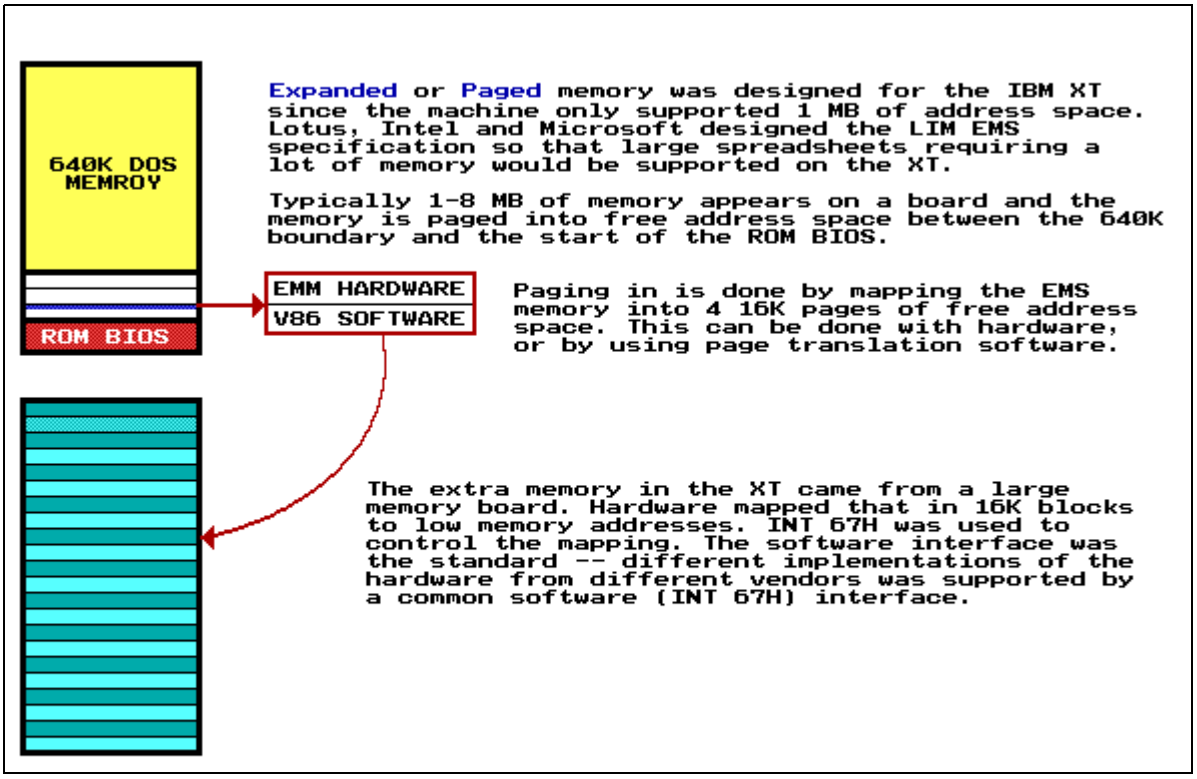


Figure 11. Paged Memory Example

Addresses not shadowed, which are in the range C0000 to FFFFF are directed to the ISA bus. Any addresses on the ISA bus which are below E0000 may be used as memory apertures.

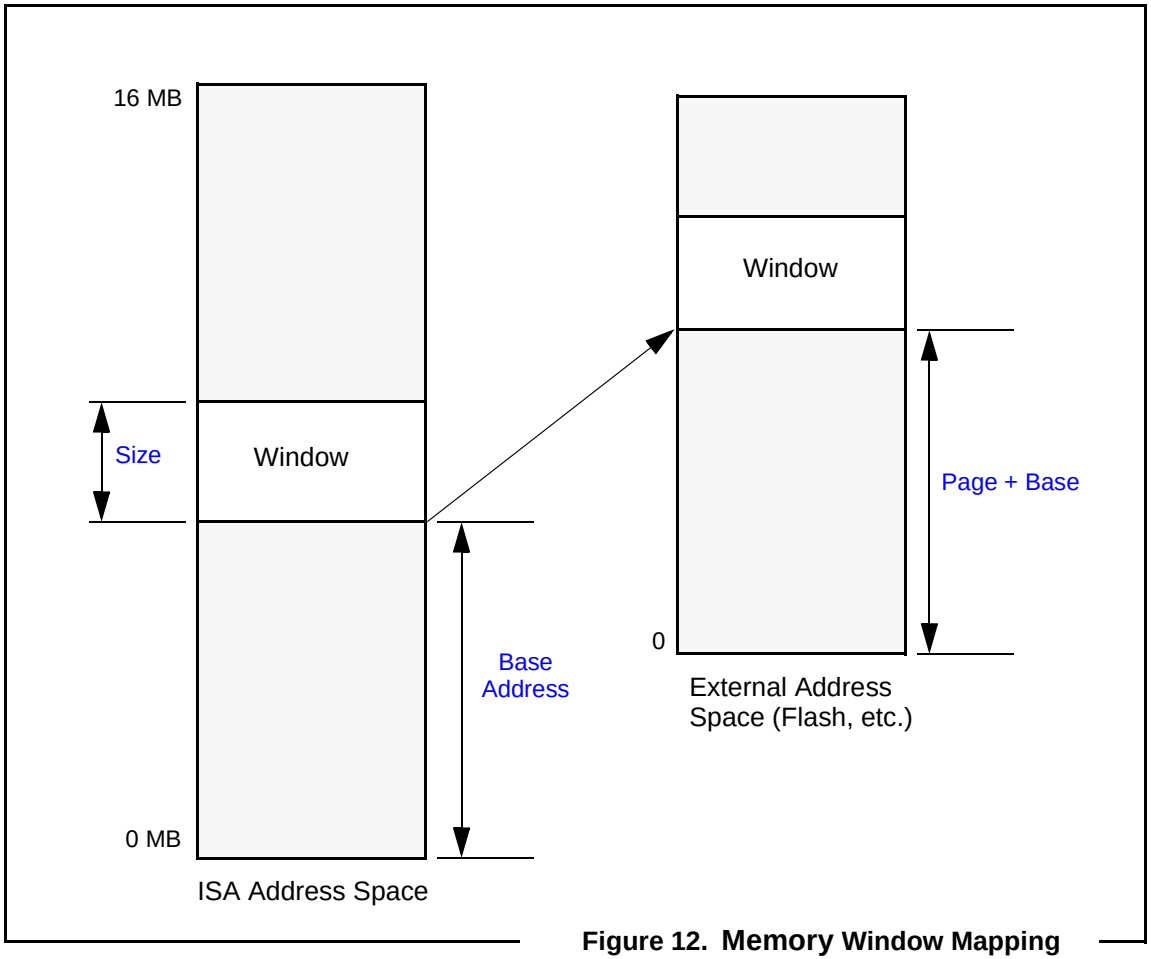


Figure 12. Memory Window Mapping

To use these memory apertures (windows) to access external SRAM or FLASH chips, the MachZ allows you to activate one of four mem_csn (n=0,1,2,3) pins when addresses between base and base+size are encountered.

Thus if you set base to D0000 and size to 64K (and those addresses were available), whenever you did a memory read or a memory write in the range of D0000 to DFFFF you would generate a memory chip select. Which chip select would you get? Well, there are

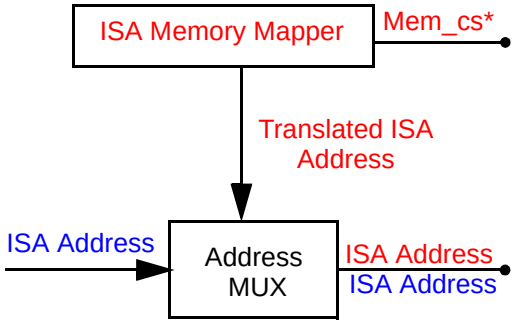


Table 12: Memory Mapper Pins

PKG	Name	Description
B04	Mem_cs0	ZF-Logic Memory Mapper CS 0
D05	Mem_cs1	ZF-Logic Memory Mapper CS 1
A03	Mem_cs2	ZF-Logic Memory Mapper CS 2
C04	Mem_cs3	ZF-Logic Memory Mapper CS 3

four register sets, one for each of the four mem_csn signals.

The address which comes out on the ISA bus is translated by the memory window hardware. In general, the question is: how do I move the target window through my flash chip so using the fixed aperture I have set up in the MachZ address space, I can reference all of the data in the flash chip (up to 16 MB). The answer is embodied in a formula for setting the value of the PAGE register:

PAGE = 1000000 - BASE + FLASHA.

If BASE = 0D0000 set PAGE to F30000 so that D000:0 goes to address 0 in the Flash. That is 1000000 - D0000 + 0.

For D0000 to go to D0000 in the flash, set PAGE to 0. That is 1000000 - D0000 + D0000. (You only specify 6 digits).

6.1.2. Using BIOS to Set Initial Memory Window Positions

AMDFLASH contains a little calculator to make the initial setup of your windows easy. While we could have built the calculator into the BIOS, we decided to have you set up the initial values of the BASE/SIZE/PAGE registers with exactly the same technique you would use later on in your own software to reset or move these memory windows.

The operative part of the BASE/SIZE/PAGE registers is a 12 bit field (3 hex digits) out of a 32-bit register:

The BASE/SIZE/PAGE registers are 32-bits wide but only bits 23-12 are used. This is 12 bits or 3 hex digits. Possible ranges of data written to these

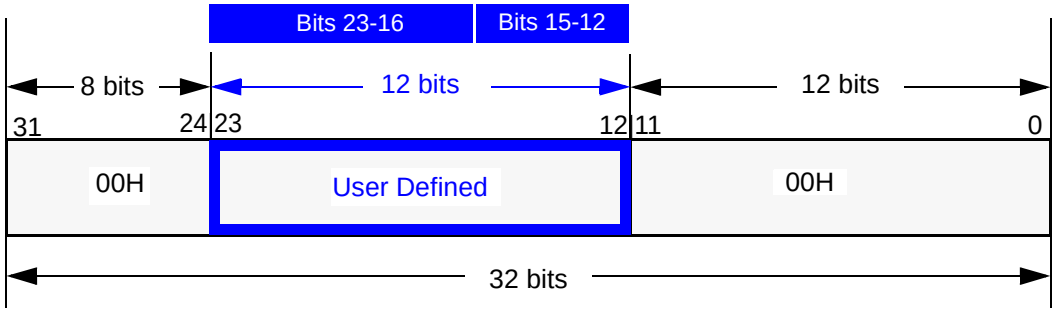


Figure 13. Fields in 32-bit memory settings register

Since each field is 12 bits wide, the actual data values are 000 to FFF hex, or 0 - 4095 decimal (in the 12 bit field).

The BASE field is easy. If you want the window to start at D0000 in the MachZ address space, the base is set to **0D0**000 hex. The SIZE and PAGE fields are harder to calculate. The aperture size is really BASE to BASE+SIZE where we assume that the right most 12 bits of size are implicitly FFF. So a size of **001000** hex would really be 1FFF providing an 8K window. You would think that therefore 0 would provide a 4K window, but we decode 0 internally to disable the aperture.

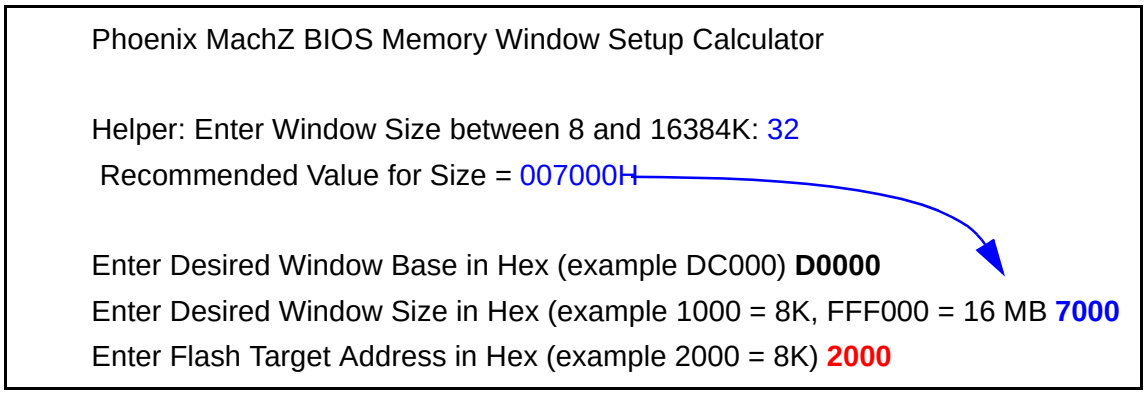
The PAGE field is almost easy. However, since it is treated as a signed number (if PAGE = -BASE then the target is the beginning of the flash), and since we don't have all 32 bits, it becomes a bit tricky.

It is best to use the calculator built into AMDFLASH. Here is an example of the printout and a copy of the source code in C:

Sample Code for Initial Memory Window Positions

Here we will create a **32K** window starting at **D0000** in the MachZ address space. The window initially point to **offset 2000H** within the flash chip.

Example: You would have to set the Page register to 00F32000H (or F32 in the MachZ Phoenix BIOS Memory Window Setup Screen).




```
1       unsigned long ulBase, ulSize, ulTarget, ulPage, ulDesiredK;
2        printf ("\r\n\n\nPhoenix MachZ BIOS Memory Window Setup Calculator \r\n", uiWorkingCS);
3
4        printf ("\r\nHelper: Enter Window Size between 8 and 16384K: ");
5        ulDesiredK = uiScanInDecimal(0);
6
7        if ((ulDesiredK >= 8) && (ulDesiredK <= 16384))
8                printf ("\r\n Recommended Value for ""Size"" = %06lXH\r\n", (ulDesiredK-4) * 1024 );
9
10        printf ("\r\nEnter Desired Window Base in Hex (example DC000) ");
11             ulBase = ulScanInHex ();
12        printf ("\r\nEnter Desired Window Size in Hex (example 1000 = 8K, FFF000 = 16 MB ");
13             ulSize = ulScanInHex ();
14             printf ("\r\nEnter Flash Target Address in Hex (example 2000 = 8K) ");
15             ulTarget = ulScanInHex ();
16                    ulBase = ulBase & 0xFFF000;
17             ulSize = ulSize & 0xFFF000 ;
18             ulTarget = ulTarget & 0xFFF000;
19             printf ("\r\nBase = %08lXH or %03lXH or %04ld decimal", ulBase, ulBase >> 12, ulBase
                        >>12);
20             if (ulSize == 0) printf ("\r\nWindow Disabled Size == 0");
21             else
```



```
22          printf ("\r\nSize = %08lXH or %03lXH or %04ld decimal for Size = %dK", ulSize, ulSize >>
                12, ulSize >>12, ((ulSize >> 12) + 1) * 4);
23          ulPage = (0x1000000 - ulBase + ulTarget) & 0xFFF000;
24          printf ("\r\nPage = %08lXH or %03lXH or %04ld decimal for Flash Offset = %08lXH\r\n\n",
25              ulPage, ulPage >> 12, ulPage >>12, ulTarget);
```

In the example above, to create a 32K window starting at D0000H which points initially to offset 2000H in the flash, you would use BASE 0D0H, SIZE 007H, and PAGE F32H.

In the 1.00 version of the BIOS these numbers need to be put in in decimal, so the values would be 208, 7, and 3890.

Here are some typical target calculations:

The table below shows the values that are used to setup a Memory Window Chip Select that enables blocks placed at flash address 0 to be mapped to ISA addresses shown in the first column.

Table 13: Sample Window Calculations

ISA Window Base	BASE (decimal)	PAGE (decimal)
C8000	200	3896
CC000	204	3892
D0000	208	3888
D4000	212	3884
D8000	216	3880
DC000	220	3876

6.2. Using the Watchdog TImer

The MachZ's Watch Dog Timer function is implemented using the following interface:

INT 15H System Services Interrupt
Calling Parameters:

Register AH = C3H Enable/Disable Watchdog Timer
Register AL = 00 Disable Watchdog Timeout

Register AL = 01 Enable Watchdog Timeout
Register BX = 1 - 255 seconds

Return Parameters:

Register Flag CF (Carry Flag) = 0 = Operation Complete
Register Flag CF (Carry Flag) = 1 = Operation Failed

WDTON.EXE is a sample test program for WDT functions. WDTON.c is the source code written in MS C and may be used as a tutorial for developing user based WDT routines in Assembler or Other C language compilers.

As an example, set the IDS board's watchdog Jumper J2 to short pin 2 to 3. (WD OSC INT). Then, at a DOS prompt, run WDTON 10. The WDTON utility will display the time continuously for 10 seconds. The test will keep the MachZ's watchdog timer from firing until the ten second interval. At that time the watchdog timer will fire and the system will reset.

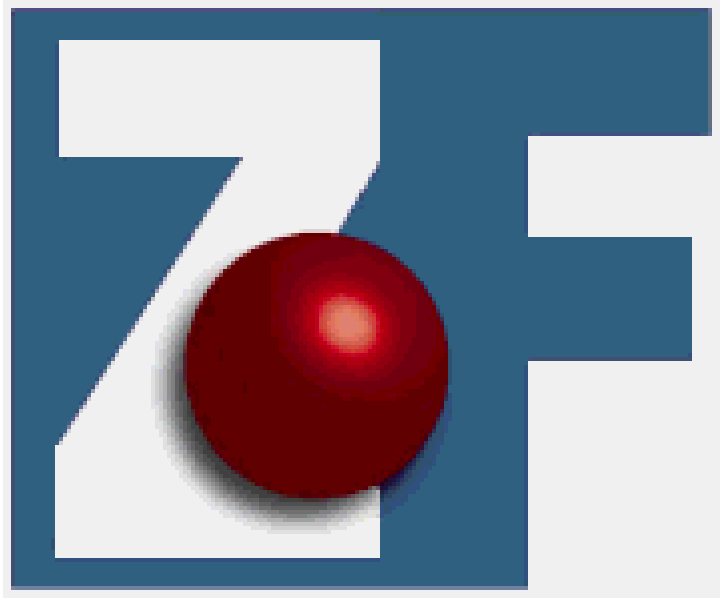
The WDTON utility and source code may obtained from us by asking support for Application Note ZFAN-013.

6.2.1. WDTON.C Sample Program

```
1  // Tests WDT INT 15 Functions
2
3  #include <ctype.h>
4  #include <io.h>
5  #include <dos.h>
6  #include <stdlib.h>
7  #include <stdio.h>
```

```
8  #include <string.h>
9  #include <time.h>
10
11  void main(argc,argv)
12  int argc;
13  char *argv[];
14  {
15  char tbuffer[9];
16  union REGS inregs, outregs;
17  int delay_value=0;
18  int wait_value=0;
19  int start_secs = 0;
20  int stop_secs = 0;
21  time_t t1,t2;
22
23  setbuf(stdout,NULL);
24
25  if(argc !=2)                        // only two args allowed
26     { printf("\nError: \n");
27       printf("Format: WDTON <secs>  i.e WDTON 10"); exit(EXIT_FAILURE);}
28
29  delay_value = (abs(atoi(argv[1])));
30  wait_value = delay_value*2;
31  system("cls");
32  //_strtime (tbuffer);
33  printf("\nWatch Dog Timer Tickle Test\n");
34  printf("\nSystem will REBOOT in %2.2d seconds",delay_value);
35  printf("\nStart Time   %s\n", _strtime (tbuffer));
36
37  //start_secs = (atoi(&tbuffer[6]));
38  start_secs = (time(&t1));
39  //printf("Startsecs == %d\n", start_secs);
40  //stop_secs =+ (atoi(&tbuffer[6]));
41  //printf("Start Secs     %d\n", start_secs);
42
43  time(&t1);
44
45  while ((difftime(t2,t1) != delay_value+1))
46     {
47     printf("Tickling WDT %s\r", _strtime (tbuffer));
48     time(&t2);
49     }
```

```
50  printf("WDT FAILED   %s\n", _strtime (tbuffer));
51  printf("\n");
52  while ((difftime(t2,t1) < delay_value+7))
53     {
54     printf("Waiting 5 more seconds: %s\r", _strtime (tbuffer));
55     time(&t2);
56     }
57  outregs.x.ax = 0xc301;
58  outregs.x.bx = delay_value;
59  int86(0x15 ,&inregs, &outregs);
60  printf("\nINT 15 function C3 01 failed CF = %d\n", outregs.x.cflag);
61  }
```



ZF Linux Devices, Inc.

1052 Elwell Court

Palo Alto, California 94303

(650) 965-3800 · Fax 965-4050

www.zflinux.com

